



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/734,551	12/11/2003	David Xinliang Li	200313032-1	2427

22879 7590 10/31/2007
HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER

YIGDALL, MICHAEL J

ART UNIT	PAPER NUMBER
----------	--------------

2192

MAIL DATE	DELIVERY MODE
-----------	---------------

10/31/2007

PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary

Application No.

10/734,551

Applicant(s)

LI ET AL.

Examiner

Michael J. Yigdoll

Art Unit

2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 21 August 2007.
- 2a) ☒ This action is FINAL. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-25 and 27 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-25 and 27 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This Office action is responsive to Applicant's submission filed on August 21, 2007.

Claims 1-25 and 27 are pending.

Response to Amendment

2. The rejection of claims 4, 6, 14 and 18 under 35 U.S.C. 112, second paragraph, has been withdrawn in view of Applicant's amendment.

3. The rejection of claims 25 and 27 under 35 U.S.C. 102(e) has been withdrawn in view of Applicant's amendment.

Response to Arguments

4. Applicant's arguments (remarks, pages 11-13) with respect to the rejection of claims 1-3, 5, 7-13, 15-17 and 19-27 under 35 U.S.C. 112, second paragraph, have been fully considered and are persuasive. The rejection has been withdrawn.

5. Applicant's arguments (remarks, page 14) with respect to the rejection of claims 1-16 under 35 U.S.C. 101 have been fully considered and are persuasive. The rejection has been withdrawn.

6. Applicant's arguments (remarks, pages 14-16) with respect to the rejection of claims 22-27 under 35 U.S.C. 101 have been fully considered but they are not persuasive.

Applicant contends that a paper-tape listing of instructions creates a functional interrelationship with the computer (remarks, page 14).

However, a paper-tape listing of instructions amounts to instructions printed on paper. Paper does not, in and of itself, create the necessary structural and functional interrelationships between the instructions and the computer that would enable any functionality to be realized. See MPEP § 2106.01.

Applicant contends that claim 23 is directed to a machine under 35 U.S.C. 101 and recites “hardware components” using means-plus-function language under 35 U.S.C. 112, and that the subject matter claimed in claim 23 does not fall within the judicially defined exceptions to patentable subject matter (remarks, page 15).

However, the claim is drafted as a system. As noted in the Office action, a reasonable interpretation of the system is that it is embodied entirely in the form of software. Applicant’s specification indicates that the “means” for performing the recited functions does not necessarily include any hardware components: “Consequently, embodiments of the invention are not limited to any one or a combination of software, firmware, hardware, or circuitry” (specification, page 20, lines 14-15). A system that is embodied entirely in the form of software amounts to descriptive material *per se* and does not fall within any category of statutory subject matter. See MPEP § 2106.01.

7. Applicant’s arguments (remarks, page 17) with respect to the rejection of claims 25 and 27 under 35 U.S.C. 102(e) have been considered but are moot in view of the new ground(s) of rejection. Applicant’s amendment necessitated the new ground(s) of rejection.

8. Applicant’s arguments (remarks, page 18-21) with respect to the rejections of claims 1-24 and 26 under 35 U.S.C. 103(a) have been fully considered but they are not persuasive.

Applicant contends that Bates fails to disclose or suggest the use of generated information mapping the clone to the function prior to link time and/or load time for use during link time and/or load time for the program (remarks, page 18).

However, Applicant is reminded that one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981), and *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). Here, the rejections are based on combinations of references including Bates, Saboff and Hunter.

Applicant contends that Saboff fails to disclose or suggest at link time for the program, if no function body of the clone is accessible by a linker, then the linker using the information mapping the clone to the function to satisfy a linker's requirement (remarks, page 18).

Specifically, Applicant contends that the interface library of Saboff does not appear to include information mapping a clone to a function (remarks, page 19).

However, as set forth in the Office action, Bates teaches a clone record that comprises information mapping a clone to a function (see, for example, column 8, line 63 to column 9, line 13). Saboff teaches a linker using an interface library to satisfy the linker's requirements (see, for example, column 6, lines 53-58). The interface library comprises mapping information (see, for example, column 5, lines 17-22). Thus, Saboff teaches a linker using mapping information to satisfy the linker's requirements. In Bates, the mapping information is included in the clone record and comprises the information mapping the clone to the function.

The test for obviousness is not that the claimed invention must be expressly suggested in any one or all of the references. Rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981). Here, the combined teachings of Bates and Saboff would have suggested the claimed subject matter to those of ordinary skill in the art.

Applicant contends that the PTO has failed to articulate a reasonable rationale for why a person of ordinary skill in the art at the time of the present invention would have been motivated to combine Bates and Saboff as asserted (remarks, page 19).

Specifically, Applicant contends that the Official Action states that a person of ordinary skill would have combined Bates and Saboff “as Saboff suggests” without identifying the suggestion of Saboff (remarks, page 19).

To the contrary, the Office action identified that the teachings of Saboff enable an application to be linked to a dynamically replaceable library that is optimized specifically for that application (see, for example, column 3, lines 10-25). Moreover, the examiner notes that a rigid application of the “teaching, suggestion, or motivation” test is not necessary to support a conclusion of obviousness. See *KSR International Co. v. Teleflex Inc.*, 550 U.S. ___, 82 USPQ2d 1385 (2007).

Applicant contends that it is incorrect that the combination of Bates and Saboff would enable the provision of a “clone in a dynamically replaceable library while still satisfying the requirements of the linker” because Saboff appears to describe replacement of an existing library

Art Unit: 2192

with an optimized version of the library without describing cloning occurring with respect to either library (remarks, pages 19-20).

However, a reasonable interpretation of Saboff is that the optimized version of the library is an optimized clone of the existing library (see, for example, column 18, line 64 to column 19, line 9). The examiner submits that the Office action establishes a *prima facie* case of obviousness.

Applicant contends that Hunter fails to disclose or suggest at load time for the program, if no function body is accessible by a loader, then the loader, based on the information mapping the clone to the function, resolving a function reference to a body of the function, and if the body of the clone is accessible by the loader, then the loader resolving the function reference to the body of the clone (remarks, page 20).

Specifically, Applicant contends that Hunter fails to disclose or suggest the use of clones of functions (remarks, page 20).

However, as set forth in the Office action, Hunter teaches creating a clone of a code sequence (see, for example, column 2, lines 24-31). Bates teaches the use of clones of functions (see, for example, the abstract).

Applicant contends that the PTO asserts that the combination of Bates and Hunter would enable "Bates to fall back to the function as a default if the clone is not accessible when the program is loaded" without identifying a teaching in Hunter or articulating a reasonable rationale for the asserted combination (remarks, page 20).

To the contrary, as identified in the Office action, Hunter teaches enabling a program to fall back to the code sequence as a default if the clone is not accessible when the program is loaded (see, for example, column 2, lines 32-49). Hunter expressly characterizes this as an “advantageous aspect of the present invention” (column 2, line 41), and indeed it is a reason for combining the teachings of Bates and Hunter.

Claim Rejections - 35 USC § 101

9. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

10. Claims 22-25 and 27 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

With respect to claim 22, the claim is directed to a method “being implemented as program instructions stored in a computer-readable medium.” However, as defined in Applicant’s specification, the computer-readable medium is not limited to statutory subject matter. Applicant defines the computer-readable medium to include, for example, “paper-tape” (page 20, line 20). In this case, the claim amounts to a mere instruction listing *per se*, which does not permit the functionality of the instructions to be realized. See MPEP § 2106.01.

With respect to claim 23, the claim is directed to a “system for using a clone cloned from a function in a program.” However, the system is not limited to statutory subject matter. In view of Applicant’s specification, a reasonable interpretation of the system is that it is embodied entirely in the form of software (see, for example, page 20, lines 14-15). In this case, the system

Art Unit: 2192

amounts to descriptive material *per se*. The claim does not recite any hardware components that would permit the functionality of the software to be realized. See MPEP § 2106.01.

With respect to claim 24, the claim does not remedy claim 23 with respect to the issue of non-statutory subject matter.

With respect to claim 25, the claim is directed to a “computer-readable medium embodying instructions for performing a method for improving performance of a program.” However, as defined in Applicant’s specification, the computer-readable medium is not limited to statutory subject matter. Applicant defines the computer-readable medium to include, for example, “paper-tape” (page 20, line 20). In this case, the claim amounts to a mere instruction listing *per se*, which does not permit the functionality of the instructions to be realized. See MPEP § 2106.01.

With respect to claim 27, the claim does not remedy claim 25 with respect to the issue of non-statutory subject matter.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

Art Unit: 2192

12. Claims 1-16, 21, 22, 25 and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,895,580 to Bates et al. (art of record, "Bates") in view of U.S. Patent No. 6,202,205 to Saboff et al. (art of record, "Saboff") and in view of U.S. Patent No. 5,920,721 to Hunter et al. (art of record, "Hunter").

With respect to claim 1 (currently amended), Bates discloses a method for improving performance of a program (see, for example, the abstract), comprising:

providing a call to a clone of a function from which the clone is created (see, for example, column 11, lines 20-23, which shows providing a call to a clone of a routine or function); the function representing programming code performing a task for the program (see, for example, column 4, lines 48-55, which shows that the routine or function performs a task);

generating information mapping the clone to the function (see, for example, column 9, lines 38-44, which shows generating a clone record, and column 8, line 63 to column 9, line 13, which shows that the clone record comprises information mapping the clone to the routine or function).

Bates does not expressly disclose:

at link time for the program, if no function body of the clone is accessible by a linker, then the linker using the information mapping the clone to the function to satisfy a linker's requirement.

However, in an analogous art, Saboff discloses creating an optimized clone of an implementation library (see, for example, column 18, line 64 to column 19, line 9). Saboff further discloses a linker that uses, at link time, an interface library to satisfy the linker's requirements (see, for example, column 6, lines 53-58). The interface library comprises

Art Unit: 2192

information mapping a routine or function to an implementation library (see, for example, column 5, lines 17-22). The implementation library is not accessible to the linker at link time (see, for example, column 7, lines 19-31). This enables an application to be linked to a dynamically replaceable library that is optimized specifically for that application (see, for example, column 3, lines 10-25).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates such that, at link time for the program, if no function body of the clone is accessible by a linker, then the linker using the information mapping the clone to the function to satisfy a linker's requirement, as Saboff suggests. This would enable Bates to provide the clone in a dynamically replaceable library while still satisfying the requirements of the linker.

Bates further discloses resolving a function reference (see, for example, column 11, lines 20-24, which shows resolving a call to a routine or function), but does not expressly disclose:

at load time for the program, if no function body of the clone is accessible by a loader, then the loader, based on the information mapping the clone to the function, resolving a function reference to a body of the function; and if the body of the clone is accessible by the loader, then the loader resolving the function reference to the body of the clone.

However, in an analogous art, Hunter discloses creating a clone of a code sequence or function (see, for example, column 2, lines 24-31). Hunter further discloses a loader that allows, at load time, selection of the clone if it is accessible and selection of the code sequence or function if the clone is not accessible, based on information mapping the clone to the code sequence or function (see, for example, column 5, line 66 to column 6, line 23). This enables a

Art Unit: 2192

program to fall back to the code sequence or function as a default if the clone is not accessible when the program is loaded (see, for example, column 2, lines 32-49).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates such that at load time for the program, if no function body of the clone is accessible by a loader, then the loader, based on the information mapping the clone to the function, resolving a function reference to a body of the function, and if the body of the clone is accessible by the loader, then the loader resolving the function reference to the body of the clone, as Hunter suggests. This would enable Bates to fall back to the function as a default if the clone is not accessible when the program is loaded.

With respect to claim 2 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that a call to the function is substituted by the call to the clone of the function (see, for example, Bates, column 11, lines 20-23, which shows that a call to the routine or function is rewritten with a call to the clone).

With respect to claim 3 (original), the rejection of claim 2 is incorporated, and Bates in view of Saboff and Hunter further discloses that a compiler substitutes the call to the function by the call to the clone of the function (see, for example, Bates, column 6, lines 33-40, which shows such a compiler).

With respect to claim 4 (currently amended), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the mapping information is included in an annotation section of an object of the program (see, for example, Hunter, column 5, lines 15-

Art Unit: 2192

18, which shows that the information is included in an architecture or annotation section of the program).

With respect to claim 5 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that a compiler generates the mapping information (see, for example, Bates, column 6, lines 33-40, which shows such a compiler).

With respect to claim 6 (currently amended), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the mapping information is stored in a loadable note section for use by the loader (see, for example, Hunter, column 5, lines 15-18, which shows that the information is included in an architecture or loadable note section for the loader).

With respect to claim 7 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the body of the clone for use by the call to that clone is selected from a list of bodies based on a priority (see, for example, Bates, column 10, lines 48-52, which shows that the clone is selected based on a score or priority).

With respect to claim 8 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the function has more than one clone in the program (see, for example, Bates, column 8, lines 37-41, which shows that the routine or function has multiple clones).

With respect to claim 9 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the clone is associated with a flag identifying

Art Unit: 2192

the clone as a function clone (see, for example, Bates, column 9, lines 26-29, which shows a flag that identifies the clone as a clone of a routine or function).

With respect to claim 10 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that symbol resolution of the clone is delayed to the load time for the program based on a linkage entry provided by the linker (see, for example, Saboff, column 5, lines 43-51, which shows that symbol resolution is delayed to load time based on a linkage entry in an interface library).

With respect to claim 11 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that a name representing the clone includes one or a combination of a condition for cloning and a name representing the function (see, for example, Bates, column 9, lines 2-5, which shows that the clone record includes a name representing the routine or function, and column 9, lines 22-25, which show that the clone record includes a condition for cloning).

With respect to claim 12 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the body of the clone is included in a library for use by the program (see, for example, Saboff, column 7, lines 12-18, which shows such a library).

With respect to claim 13 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that a compiler creates the body of the clone based on a programming statement provided to the compiler (see, for example, Bates, column 6, lines

Art Unit: 2192

33-40, which shows such a compiler, and column 11, line 65 to column 12, line 5, which shows that the clone is created based on a statement provided to the compiler).

With respect to claim 14 (currently amended), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the compiler creates the body of the clone after an analysis determining advantages and disadvantages of such creation (see, for example, Bates, column 6, lines 33-40, which shows such a compiler, and column 9, lines 52-63, which shows an analysis to determine the relative value of creating the clone).

With respect to claim 15 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the clone is created based on one or a combination of:

- a logical relationship between at least two parameters passed to the function;

- an alias-relationship between at least two parameters passed to the function;

- a value of at least one parameter passed to the function from; and

- a number of alignment bytes of at least one parameter passed to the function

(see, for example, Bates, column 11, lines 45-61, which shows that the clone is created based on logical relationships among and values of parameters passed to the routine or function).

With respect to claim 16 (original), the rejection of claim 1 is incorporated, and Bates in view of Saboff and Hunter further discloses that the clone is created based on profile data of the function (see, for example, Bates, column 10, lines 26-29, which shows that the clone is created based on profile data).

With respect to claim 21 (original), Bates discloses a method for using a clone cloned from a section of code of a program (see, for example, the abstract), comprising:

substituting a call to the section of the code by a call to the clone (see, for example, column 11, lines 20-23, which shows rewriting a call to the routine with a call to the clone).

Bates discloses mapping the clone to the section of code (see, for example, column 8, line 63 to column 9, line 13, which shows a clone record that comprises information mapping the clone to the routine), but does not expressly disclose:

at link time for the program, mapping the clone to the section of code.

However, in an analogous art, Saboff discloses creating an optimized clone of an implementation library (see, for example, column 18, line 64 to column 19, line 9). Saboff further discloses a linker that uses, at link time, an interface library (see, for example, column 6, lines 53-58). The interface library comprises information mapping a routine to an implementation library (see, for example, column 5, lines 17-22), which is not accessible to the linker at link time (see, for example, column 7, lines 19-31). This enables an application to be linked to a dynamically replaceable library that is optimized specifically for that application (see, for example, column 3, lines 10-25).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates so as to, at link time for the program, map the clone to the section of code, as Saboff suggests. This would enable Bates to provide the clone in a dynamically replaceable library.

Bates does not expressly disclose:

at load time for the program, mapping the clone to the section of code; and

Art Unit: 2192

during execution of the program, if a body of the clone is available in a library used by the program, then using that body, else if the body of the clone is not available in the library, then using the section of code from which the clone is cloned.

However, in an analogous art, Hunter discloses creating a clone of a code sequence (see, for example, column 2, lines 24-31). Hunter further discloses a loader that uses, at load time, information mapping the clone to the code sequence, so as to select the clone if it is available and to select the code sequence if the clone is not available (see, for example, column 5, line 66 to column 6, line 23). This enables a program to fall back to the code sequence as a default if the clone is not available when the program is executed (see, for example, column 2, lines 32-49).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates so as to, at load time for the program, map the clone to the section of code, and during execution of the program, if a body of the clone is available in a library used by the program, then use that body, else if the body of the clone is not available in the library, then use the section of code from which the clone is cloned, as Hunter suggests. This would enable Bates to fall back to the section of code as a default if the clone is not available when the program is executed.

With respect to claim 22 (original), the rejection of claim 21 is incorporated, and Bates in view of Saboff and Hunter further discloses the method being implemented as program instructions stored in a computer-readable medium (see, for example, Bates, column 5, line 58 to column 6, line 15).

Art Unit: 2192

With respect to claim 25 (currently amended), Bates discloses a computer-readable medium embodying instructions for performing a method for improving performance of a program (see, for example, the abstract), the method comprising:

providing a call to a clone of a function from which the clone is created (see, for example, column 11, lines 20-23, which shows providing a call to a clone of a routine or function); the function representing programming code performing a task for the program (see, for example, column 4, lines 48-55, which shows that the routine or function performs a task);

generating information mapping the clone to the function (see, for example, column 9, lines 38-44, which shows generating a clone record, and column 8, line 63 to column 9, line 13, which shows that the clone record comprises information mapping the clone to the routine or function); and

creating the clone based on one or a combination of

a logical relationship between at least two parameters passed to the function;

an alias-relationship between at least two parameters passed to the function;

a value of at least one parameter passed to the function from; and

a number of alignment bytes of at least one parameter passed to the function

(see, for example, column 11, lines 45-61, which shows that the clone is created based on logical relationships among and values of parameters passed to the routine or function).

Bates does not expressly disclose:

at link time for the program, if no function body of the clone is accessible by a linker, then the linker using the information mapping the clone to the function to satisfy a linker's requirement.

However, in an analogous art, Saboff discloses creating an optimized clone of an implementation library (see, for example, column 18, line 64 to column 19, line 9). Saboff further discloses a linker that uses, at link time, an interface library to satisfy the linker's requirements (see, for example, column 6, lines 53-58). The interface library comprises information mapping a routine or function to an implementation library (see, for example, column 5, lines 17-22). The implementation library is not accessible to the linker at link time (see, for example, column 7, lines 19-31). This enables an application to be linked to a dynamically replaceable library that is optimized specifically for that application (see, for example, column 3, lines 10-25).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates such that at link time for the program, if no function body of the clone is accessible by a linker, then the linker using the information mapping the clone to the function to satisfy a linker's requirement, as Saboff suggests. This would enable Bates to provide the clone in a dynamically replaceable library while still satisfying the requirements of the linker.

Bates further discloses resolving a function reference (see, for example, column 11, lines 20-24, which shows resolving a call to a routine or function), but does not expressly disclose:

at load time for the program, if no function body of the clone is accessible by a loader, then the loader, based on the information mapping the clone to the function, resolving a function reference a body of the function; and if the body of the clone is accessible by the loader, then the loader resolving a function reference to the body of the clone.

However, in an analogous art, Hunter discloses creating a clone of a code sequence or function (see, for example, column 2, lines 24-31). Hunter further discloses a loader that allows, at load time, selection of the clone if it is accessible and selection of the code sequence or function if the clone is not accessible, based on information mapping the clone to the code sequence or function (see, for example, column 5, line 66 to column 6, line 23). This enables a program to fall back to the code sequence or function as a default if the clone is not accessible when the program is loaded (see, for example, column 2, lines 32-49).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates such that at load time for the program, if no function body of the clone is accessible by a loader, then the loader, based on the information mapping the clone to the function, resolving a function reference to a body of the function, and if the body of the clone is accessible by the loader, then the loader resolving a function reference to the body of the clone, as Hunter suggests. This would enable Bates to fall back to the function as a default if the clone is not accessible when the program is loaded.

With respect to claim 27 (original), the rejection of claim 25 is incorporated, and Bates in view of Saboff and Hunter further discloses that the program includes multiple calls to multiple clones (see, for example, Bates, column 8, lines 37-41, which shows that the program has multiple calls to multiple clones).

13. Claims 17-20, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bates in view of Saboff.

With respect to claim 17 (original), Bates discloses a method for using a clone cloned from a function in a program (see, for example, the abstract).

Bates discloses information mapping the clone to the function (see, for example, column 8, line 63 to column 9, line 13, which shows a clone record that comprises information mapping the clone to the routine or function), and further discloses a call to the clone (see, for example, column 11, lines 20-23, which shows a call to a clone of a routine or function), but does not expressly disclose:

using information mapping the clone to the function to satisfy a linker's requirement of having a clone body for a call to the clone; the linker's requirement being part of building the program; and

building a library that includes the body of the clone.

However, in an analogous art, Saboff discloses building an optimized clone of an implementation library (see, for example, column 18, line 64 to column 19, line 9). Saboff further discloses a linker that uses, at link time, an interface library to satisfy the linker's requirements when building the program (see, for example, column 6, lines 53-58). The interface library comprises information mapping a routine or function to an implementation library (see, for example, column 5, lines 17-22). The implementation library is not accessible to the linker at link time (see, for example, column 7, lines 19-31). This enables an application to be linked to a dynamically replaceable library that is optimized specifically for that application (see, for example, column 3, lines 10-25).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Bates so as to use information mapping the clone to the

Art Unit: 2192

function to satisfy a linker's requirement of having a clone body for a call to the clone, the linker's requirement being part of building the program, and to build a library that includes the body of the clone, as Saboff suggests. This would enable Bates to provide the clone in a dynamically replaceable library while still satisfying the requirements of the linker.

Bates in view of Saboff further discloses:

wherein the function represents programming code performing a task for the program (see, for example, Bates, column 4, lines 48-55, which shows that the routine or function performs a task) and building the program and the library are independent of one another (see, for example, Saboff, column 19, lines 18-25, which shows that the program the library are built independently).

With respect to claim 18 (currently amended), the rejection of claim 17 is incorporated, and Bates in view of Saboff further discloses, prior to building the library that includes the body of the clone, building a library that does not include the body of the clone (see, for example, Saboff, column 18, line 64 to column 19, line 9; which shows building an unoptimized library that does not include the body of the clone).

With respect to claim 19 (original), the rejection of claim 17 is incorporated, and Bates in view of Saboff further discloses that the call to the clone has replaced a call to the function (see, for example, Bates, column 11, lines 20-23, which shows that a call to the routine or function is rewritten with a call to the clone).

With respect to claim 20 (original), the rejection of claim 17 is incorporated, and Bates in view of Saboff further discloses that the clone is created based on information passed to the

Art Unit: 2192

function (see, for example, Bates, column 11, lines 45-61, which shows that the clone is created based on information passed to the routine or function).

With respect to claim 23 (original), the claim is directed to a system that corresponds to the method of claim 17 (see the rejection of claim 17 above).

With respect to claim 24 (original), the claim is directed to a system that corresponds to the method of claim 20 (see the rejection of claim 20 above).

Conclusion

14. Applicant's amendment necessitated any new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Art Unit: 2192

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy



TUAN DAM
SUPERVISORY PATENT EXAMINER